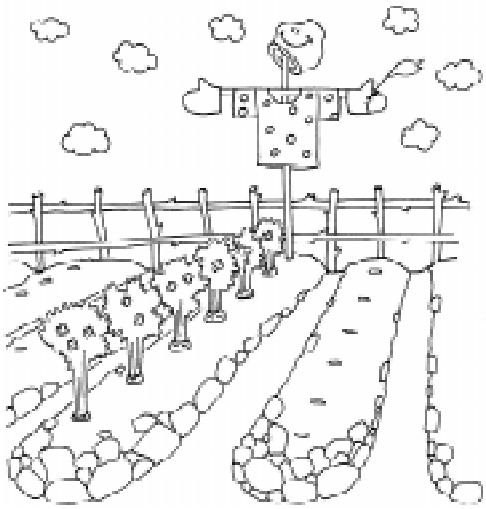
Школа современного программирования

Романовский Иосиф Владимирович,
Дьяченко Василий Владимирович

ДЕМОНСТРАЦИОННЫЕ ПРОГРАММЫ: 2. СУФФИКСНЫЙ МАССИВ

Суффиксные массивы служат для решения многих задач со строками, например, задачи о подстроке [1]. Многие прикладные задачи (например, в генетике) требуют для своего решения многократный поиск в одной и той же строке. Выгодно провести предварительную обработку (препроцессинг) строки, то есть обработать её таким образом, чтобы потом поиск одного образца выполнялся как можно быстрее. Если образцов мало, это невыгодно и лучше пользоваться другими алгоритмами точного поиска в строках. Но чем больше образцов мы проверяем, тем препроцессинг выгоднее использовать.



В конец строки для удобства поиска вставляется специальный символ...

В 1993 году У. Манбер и Дж. Майерс [2] предложили для решения задачи о подстроке структуру, названную суффиксным массивом. В книге [1] описан метод построения этого массива с помощью суффиксного дерева, который очень громоздок. В последнее время появилось много замечательных методов. Здесь описывается один из них [3, см. также 4]. О быстродействии алгоритмов можно судить по тому, что суффиксный массив для полного текста английского перевода Библии (4 мегабайта при 63 различных символах) строится ими за 1,5–2 секунды, а человеческая хромосома № 22 (32 мегабайта при 5 различных символах) – за 16 – 40 секунд (на компьютере с 512 MB памяти и процессором Интел 1,3 GHz под операционной системой Linux).

1. СУФФИКСНЫЙ МАССИВ

Пусть задана m-символьная строка T. *Суффиксным массивом* для T, обозначенным Pos, называется массив целых чисел от 1 до m, определяющих лексикографический порядок всех m суффиксов строки.

В конец строки для удобства поиска вставляется специальный символ, больше нигде в тексте не встречающийся. Он называется «сентинел»[1] и обычно изобража-

[1] *Sentinel* (англ.) — отдельно выставленный часовой, часто обозначающий границу охраняемой территории. В русском языке ему может соответствовать слово «флажковый». Именно флажковыми называются военнослужащие с флажками в начале и конце движущейся колонны.

```
Pos[ 1] = 1  (aaddaaaddadadaaa$)
Pos[ 2] = 2  (addaaaddadadaaa$)
Pos[ 3] = 3  (ddaaaddadadaaa$)
Pos[ 4] = 4  (daaaddadadaaa$)
Pos[ 5] = 5  (aaaddadadaaa$)
Pos[ 6] = 6  (aaddadadaaa$)
Pos[ 7] = 7  (addadadaaa$)
Pos[ 8] = 8  (ddadadaaa$)
Pos[ 9] = 9  (dadadaaa$)
Pos[10] = 10 (adadaaa$)
Pos[11] = 11 (dadaaa$)
Pos[12] = 12 (adaaa$)
Pos[13] = 13 (daaa$)
Pos[14] = 14 (aaa$)
Pos[15] = 15 (aa$)
Pos[16] = 16 (a$)
Pos[17] = 17 ($)
```

Рис. 1

```
Pos[ 1] = 17 ($)
Pos[ 2] = 16 (a$)
Pos[ 3] = 15 (aa$)
Pos[ 4] = 14 (aaa$)
Pos[ 5] = 5  (aaaddadadaaa$)
Pos[ 6] = 1  (aaddaaaddadadaaa$)
Pos[ 7] = 6  (aaddadadaaa$)
Pos[ 8] = 12 (adaaa$)
Pos[ 9] = 10 (adadaaa$)
Pos[10] = 2  (addaaaddadadaaa$)
Pos[11] = 7  (addadadaaa$)
Pos[12] = 13 (daaa$)
Pos[13] = 4  (daaaddadadaaa$)
Pos[14] = 11 (dadaaa$)
Pos[15] = 9  (dadadaaa$)
Pos[16] = 3  (ddaaaddadadaaa$)
Pos[17] = 8  (ddadadaaa$)
```

Рис. 2

ется знаком «$». Этот символ лексикографически предшествует всем остальным.

Каждый суффикс строки определяется своим номером. Первый суффикс – это сама строка, последний суффикс – последняя буква строки. Для хранения суффиксов нам надо хранить только их номера. В массиве Pos хранятся номера суффиксов в таком порядке, что суффиксы с этими номерами расположены лексикографически. Так как в суффиксном массиве Pos хранятся только целые числа, а не строки, он не занимает много памяти.

В начале алгоритма суффиксный массив выглядит так (см. рис. 1).

В соответствии с поставленной задачей, для строки «aaddaaaddadadaaa$» суффиксный массив должен выглядеть следующим образом (см. рис. 2).

Использовать такой массив для поиска подстроки не сложно. Пусть надо найти вхождения строки «ad» в строку, суффиксный массив которой расположен выше. Для этого достаточно найти номер первого суффикса начинающегося на «ad». Это несложно сделать двоичным поиском. Когда искомый номер найден (в данном случае это 8), все остальные суффиксы находятся рядом с ним в суффиксном массиве и имеют номера 9 10 11.

Таким образом, ответ: 12 10 2 7 – это и есть вхождения подстроки «ad» в строку «aaddaaaddadadaaa$»

... корзины — множества суффиксов, у которых первые несколько букв совпадают.

2. ПРИНЦИП ПОСТРОЕНИЯ СУФФИКСНОГО МАССИВА

Существует множество алгоритмов построения суффиксного массива, однако принцип работы у них почти одинаков. Определяются так называемые *корзины* – множества суффиксов, у которых первые несколько букв совпадают. Каждая корзина маркируется номером первого суффикса, входящего в упорядочение, и этот номер одинаков для всех суффиксов корзины.

Вот, например, как представляется в демонстрационной программе корзина суффиксов с первыми двумя одинаковыми буквами «aa»: Она имеет номер 3, так как перед этими суффиксами стоят суффиксы «$» и «a$» (рис. 3).

3	a	a	$													
3	a	a	a	$												
3	a	a	d	d	a	d	a	d	a	a	a	$				
3	a	a	a	d	d	a	d	a	d	a	a	a	$			
3	a	a	d	d	a	a	a	d	d	a	d	a	d	a	a	a	$

Рис. 3

Суть алгоритмов в том, что суффиксы разбиваются по корзинам, а затем эти корзины уточняются (мы усиливаем условие нахождения двух суффиксов в одной корзине и таким образом разбиваем корзины на более мелкие). Процесс продолжается, пока в каждой корзине не останется только один суффикс.

3. АЛГОРИТМ

Теперь рассмотрим алгоритм построения массива, реализованный в демонстрационной программе.

Фаза первая

На первой фазе мы грубо раскидываем суффиксы по корзинам, обращая внимание только на первые две буквы суффиксов. (стоит отметить что здесь и далее по тексту операции совершаются не над суффиксами а над числами в суффиксном массиве, соответствующими этим суффиксам) Для этого достаточно сделать два сканирования по последовательности.

При первом сканировании мы вычисляем ключи сортировки для суффиксов. Ключи сортировки вычисляется из соображений лексикографической упорядоченности. Также при первом сканировании мы считаем статистику появления пар букв в строке.

После первого сканирования место под корзины распределяем согласно собранной статистике.

При втором сканировании просто пробегаемся по всем суффиксам и помещаем их по корзинам в соответствии с найденными ранее ключами сортировки.

Фаза вторая

После первой фазы мы имеем массив, в котором суффиксы лексикографически упорядочены по первым двум буквам. При этом, чтобы этот массив удовлетворял необходимому условию, нужно разобраться с суффиксами, для которых упорядочения по двум буквам не хватило. Такие суффиксы лежат с «конфликтующими» в одной корзине.

Вот пример работы демонстрационной программы для той же строки «aaddaaaddadadaaa$» после первой фазы (рис. 4).

Как видно из рисунка, необходимо уточнить корзину под номером 3 (которая приводилась в примере выше).

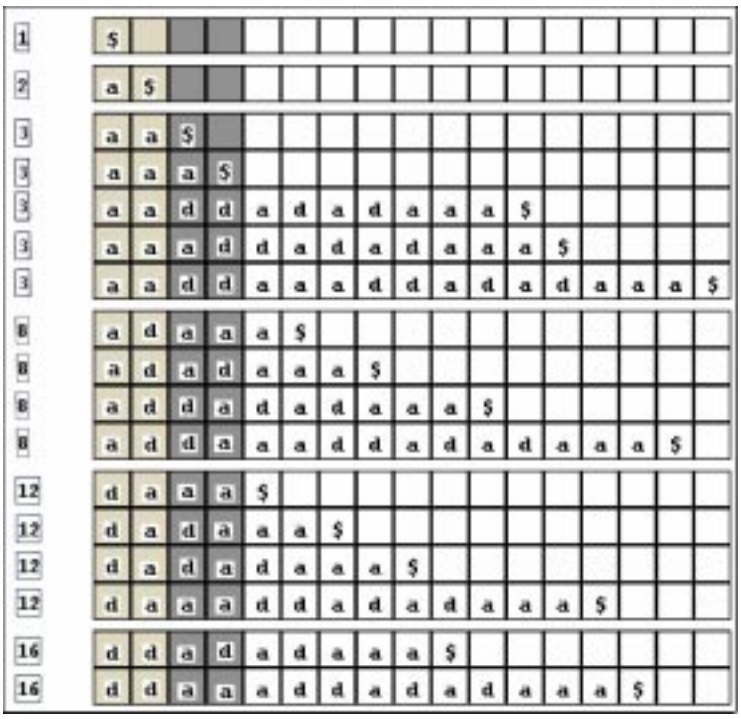


Рис. 4

Уточнение корзины происходит следующим образом: сначала пробегаемся по суффиксам корзины и вычисляем ключи сортировки для суффиксов. Ключи сортировки на этот раз вычисляются из соображения что суффиксы уже отсортированы по первым двум буквам, а именно «на две буквы» мы и собираемся уточнить корзину. Например, для суффикса «aaddadadaaa$» ключ сортировки будет равен номеру корзины, в которой находится часть, по которой будет идти сравнение («ddadadaaa$»). В данном случае это 16 (см. рис. 5).

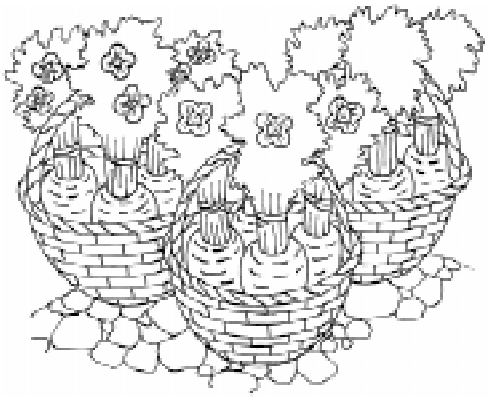
Для суффикса «aaaddadadaaa$» по аналогии, ключ его сортировки равен номеру корзины суффикса «addadadaaa$», то есть 8 (рис. 6).

После того как вычислены все ключи сортировки, просто сортируем суффиксы в корзине согласно этим ключам. На практике используется вставка (для корзин меньше 15-ти суффиксов) и Quicksort. Сразу после сортировки корзина разбивается на подкорзины меньшего размера.

В нашем примере, все же остается одна неразбитая корзина, которую мы уточним при следующем повторении второй фазы (рис. 7).

Аналогично разбиваем оставшиеся корзины. Здесь стоит отметить, что чем позднее стоит корзина, тем «лучше» будет в ней разбиение. Например, при уточнении 12-ой корзины, суффикс «daaa$» получит ключ сортировки отличный от «daaaddadadaaa$» благодаря тому, что корзина 3, ранее состоявшая из пяти суффиксов, была разбита. Таким образом, мы используем результаты разбиения на корзины сразу же после этого разбиения.

После прохода по всем корзинам, имеем массив суффиксов, упорядоченных по первым четырем буквам. Если этого недостаточно (есть хотя бы одна корзина с двумя суффиксами) – повторяем вторую фазу, но уже пользуясь тем что суффиксы упорядочены по большему числу букв (4). Если и после этого пробега останутся неразбитые корзины, снова повторяем вторую фазу с условием упорядоченности по первым восьми буквам. И так далее по степеням двойки до тех пор пока не получим по одному суффиксу в одной корзине.



...после сортировки корзина разбивается на подкорзины меньшего размера.

То, что получится после многократного выполнения второй фазы, и будет суффиксным массивом.

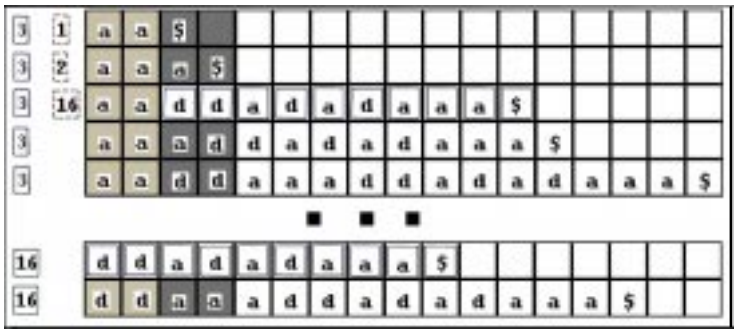


Рис. 5

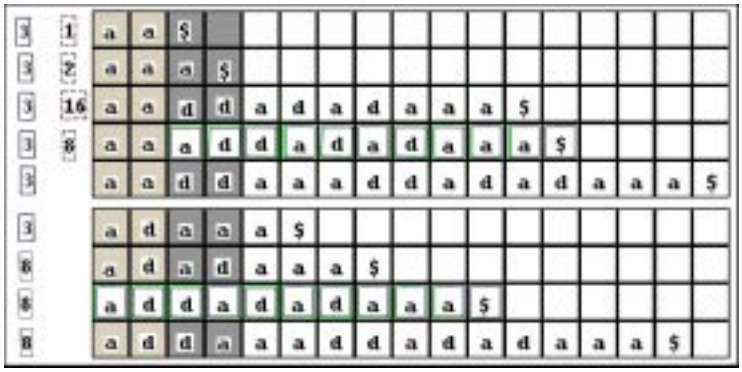


Рис. 6

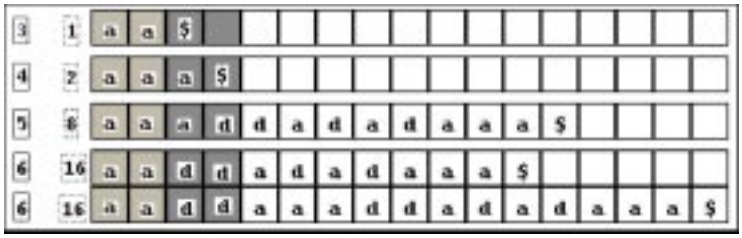


Рис. 7

4. ИНТЕРФЕЙС

Для запуска демонстрации необходимо нажать кнопку «воспроизведения» в нижней правой части экрана. Для лучшего понимания алгоритма рекомендуется нажать на кнопку «пауза» в правом нижнем углу и смотреть демонстрацию по шагам, читая разъясняющие комментарии для каждого шага (рис. 8).

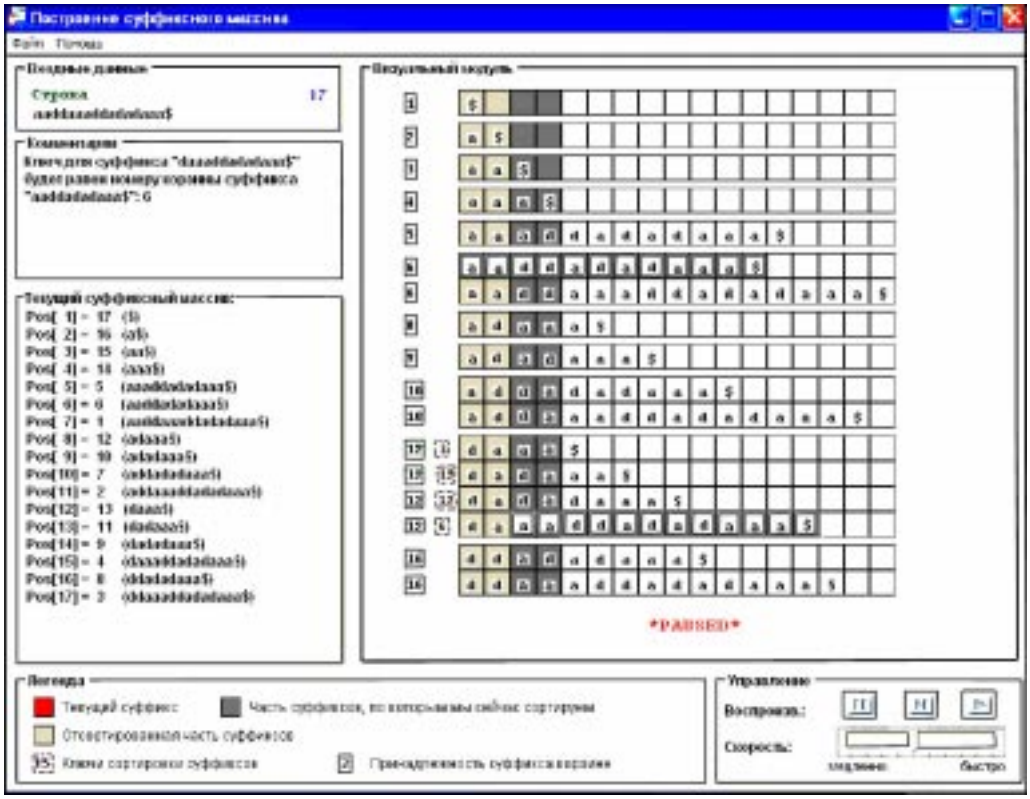


Рис. 8

Литература

1. *Гасфилд Дэн.* Строки, деревья и последовательности в алгоритмах. СПб.: «Невский диалект», БХВ-Петербург, 2003. 654 с.

2. *Manber U., Myers G.* Suffix arrays, pages a new method for on-line search // SIAM J. Comput., 22, 1993. P. 935–948.

3. *Schurmann K.-B., Stoye J.* An incomplex algorithm for fast suffix array construction // Software, Pract. Exper. 37(3), 2007. P. 309–329.

4. *Bentley J.L., Mellroy M.D.* Engineering a sort function // Software, Pract.Exper. 22(11), 1993. P. 1249–1265.

Романовский Иосиф Владимирович, доктор физико-математических наук, профессор СПбГУ,

Дьяченко Василий Владимирович, студент математико-механического факультета СПбГУ.